TEXT DETECTION AND EXTRACTION FROM DIGITAL IMAGE

INSTITUTE OF AERONAUTICAL ENGINEERING

NAME	:	NNV SAI DEEPAK
ROLL NO.	:	19951A04C2
BRANCH	:	ECE
SECTION	:	B
COURSE NAME	:	PROECT BASED LEARNING
COURSE CODE	:	AHSB15

ABSTRACT:

It is easy for humans to understand the contents of an image by just looking at it. You can recognize the text on the image and can understand it without much difficulty. However, computers don't function similarly. They only understand information that is organized. And this is exactly where Optical Character Recognition comes in the picture. This project majorly focuses on the OCR's application areas using Tesseract OCR, OpenCV, installation & environment setup, coding, and limitations of Tesseract.

KEYWORDS: Text detection, text extraction

INTRODUCTION:

In recent years, with the development of multimedia technologies and the Internet, the number of digital images transferred through the Internet has increased tremendously. The text that contained in the images carries the information, which is important for fully understanding the images. If the text can be automatically detected, extracted and recognized by computers, a more reliable content-based access to the image data can be achieved, which provides a new way for the application of content-based images and videos. Therefore, how to locate and extract textual information quickly and accurately from images becomes a hot topic area in the world today.

This is a very economic technology and can be used in several other fields as well, few are listed as below:

- Banking (To read Credit Card)
- Government Sector (Form Processing)
- Used in Car Number Plate Recognition System

PROBLEM STATEMENT:

In general, to extract text from image, Image processing techniques are used. Which are a bit complex and time taking and high knowledge is required regarding MATLAB.

EXISTING MODEL:

Qi Zheng in his work tried to automatically recognize characters using segmentation algorithm with multiple-size sliding sub-windows. The customized program generates template images first and the extracted SIFT features are matched to the template images. After using the segmentation algorithm multiple single-character-areas are identified and the results are verified by a voting and geometric verification algorithm.

Masakazu Iwamura in his paper, propose a character recognition method using local features with several desirable properties. The novelty of the proposed method is to take into account the arrangement of local features so as to recognize multiple.

PROPOSED MODEL:

To reduce the complexity of code we here use python language and some libraries like OpenCV, pytesseract. By using some inbuilt functions of the imported libraries we detect and extract text from images.

Some of the functions of the libraries as follows:

- Image_to_string()
- Image_to_boxes()
- Cv2.imread()
- Cv2.imshow()

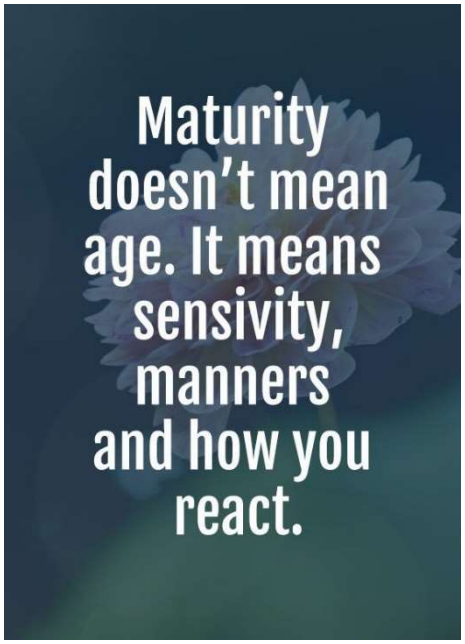
  Etc.

OUTPUT AND DISCUSSIONS:

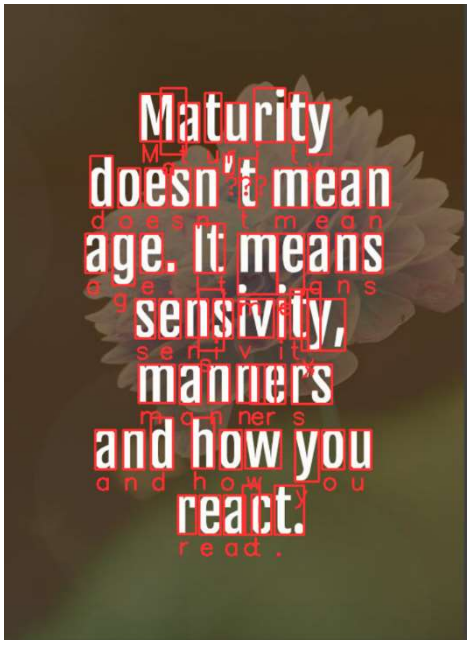
When we insert an image with text using OpenCV, the text from the image got extracted and will get displayed in the output terminal, and an image will be displayed where every character in the image get highlighted and the text appears with the image.

Example:

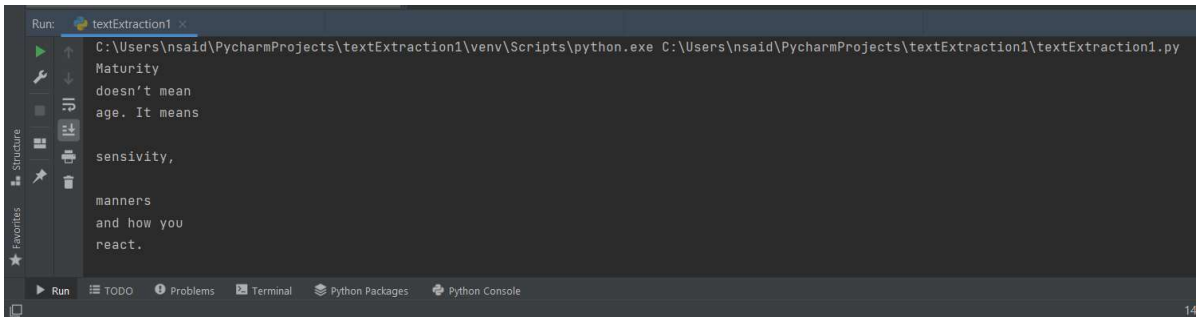
Input image:



Output image:



Output terminal:



CONCLUSION:

In the end, it can be concluded that Tesseract is perfect for scanning clean documents and you can easily convert the image's text from OCR to word, pdf to word, or to any other required format. It has pretty high accuracy and font variability. This is very useful in case of institutions where a lot of documentation is involved such as government offices, hospitals, educational institutes, etc. In the current release 4.0, Tesseract supports OCR based deep learning that is significantly more accurate.

Appendix:

Code:

```
import cv2
import pytesseract

pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files\\Tesseract-
OCR\\tesseract.exe'
img = cv2.imread('sample2.png')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
hImg, wImg,_ = img.shape
boxes = pytesseract.image_to_boxes(img)
print(pytesseract.image_to_string(img))
for b in boxes.splitlines():
    b = b.split(' ')
    x, y, w, h = int(b[1]), int(b[2]), int(b[3]), int(b[4])
    cv2.rectangle(img, (x,hImg- y), (w,hImg- h), (50, 50, 255), 2)
    cv2.putText(img,b[0],(x,hImg y+25)
,cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,255),2)
cv2.imshow('img', img)
cv2.waitKey(0)
```